CHARUSAT

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CE143: COMPUTER CONCEPTS & PROGRAMMING

Chapter – 1

Introduction to ‘C’ language

DEPSTAR

Devang Patel Institute of Advance Technology And Research

Objectives

- To get understanding of evolution of ‘C’ Language
- To develop programming skills using the fundamentals and basics of C language
- To impart the knowledge about basic structure which is the backbone of programming
- To understand the significance of ‘C’ language
- To teach the basics of preprocessors available with C compiler
- To Develop an understanding of the compilation process

1-1 History of C

- C is a programming language which born at “AT & T’s Bell Laboratory” of USA in 1972.

- C was written by **Dennis Ritchie**, that is why he is also called as father of c programming language.

- C language was created for a specific purpose i.e designing the UNIX operating system (which is currently base of many UNIX based OS).

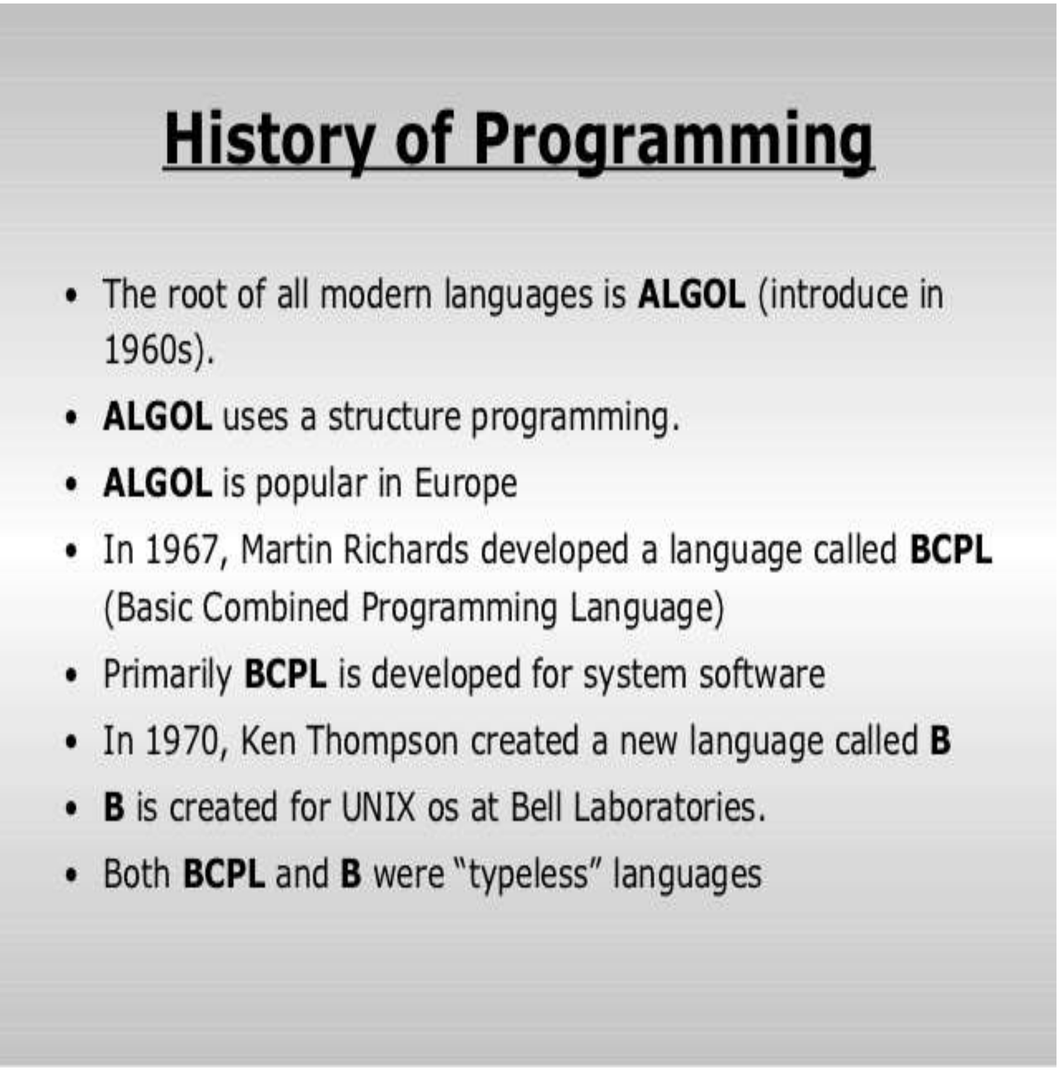
- From the beginning, C was intended to be useful to allow busy programmers to get things done because C is such a powerful and dominant language

1-1 History of C Continued…

Why Name "C" was given to Language ?

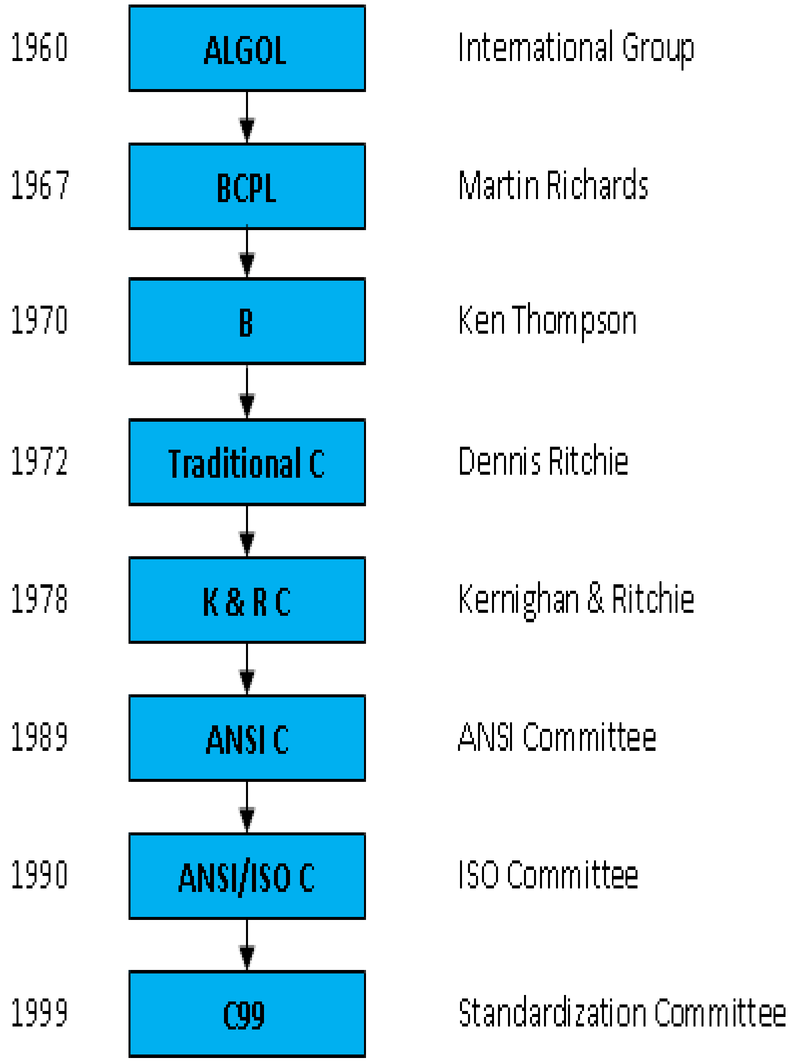
- Many of C's principles and ideas were derived from the earlier language B. (Ken Thompson was the developer of B Language.)
- BCPL and CPL are the earlier ancestors of B Language , CPL is combined Programming Language. In 1967, BCPL Language (Basic CPL) was created as a scaled down version of CPL.
- As many of the **features were derived from "B" Language that is why it was named as "C"**.
- After 7-8 years C++ came into existence which was first example of object oriented programming .

1-1 History of C Continued…



History of Programming

- The root of all modern languages is **ALGOL** (introduce in 1960s).
- **ALGOL** uses a structure programming.
- **ALGOL** is popular in Europe
- In 1967, Martin Richards developed a language called **BCPL** (Basic Combined Programming Language)
- Primarily **BCPL** is developed for system software
- In 1970, Ken Thompson created a new language called **B**
- **B** is created for UNIX os at Bell Laboratories.
- Both **BCPL** and **B** were "typeless" languages



1-2 Characteristics of C

1. **Simple and Clear**----C has richest collection of data types(int,float,char), operators(symbolic representation) and inbuilt functions.

2. **Stuctured Approach**----C provide the fundamental control flow constructions required for well structured program like statement grouping, decision making(if-else),switch-case, break etc.

3. **Portable**----C has the characteristics of portability .Portable means computer independent. The programs made in C on one computer can be run on different computers by not changing anything.

4. **Case** Sensitive----Case Sensitive means that almost everything in C program is written in small case.

1-2 Characteristics of C Continued

5. **Easily Available and Quick----**The C compiler are very easily available and require very less disk space .

TURBO C is the compiler which is very easy to load in your computer.

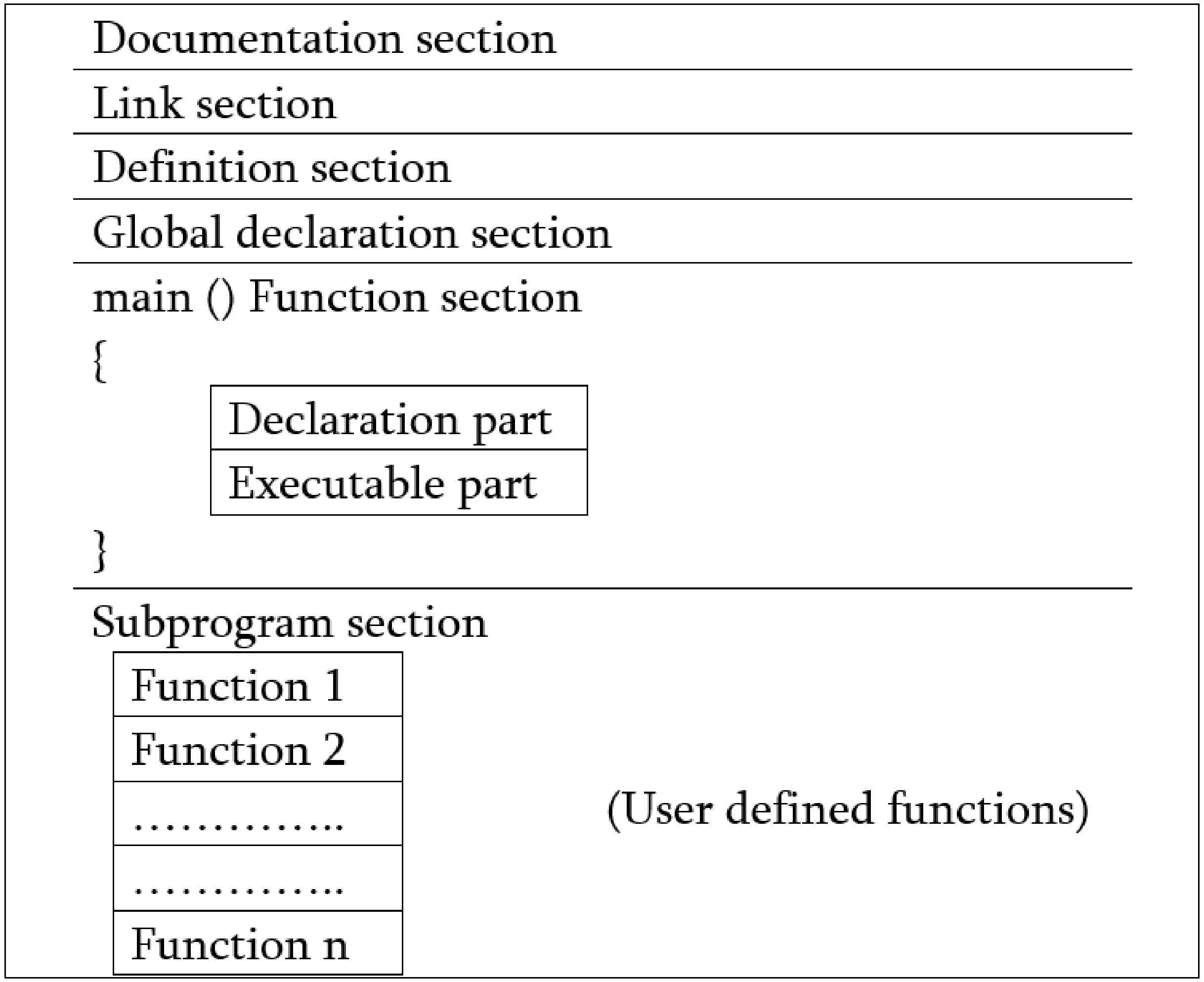
6. **Modular----**Modular means the programs can be easily divided into small modules with the use of functions.

7. **Easy Error detection** Whenever we compile the program after typing it in the editor, the errors are immediately displayed on the screen pointing to the line numbers where they are actually coming.

8. **Availability of recursive function----**This help us to reduce the writing of definition of functions by calling same function again and again reduce the lines of codes as well and are easy to remember.

1-3 Basic Structure of ‘C’ Program

- **Structure of C Program**



Documentation section

Link section

Definition section

Global declaration section

main () Function section

{

Declaration part

Executable part

}

Subprogram section

Function 1

Function 2

…………..

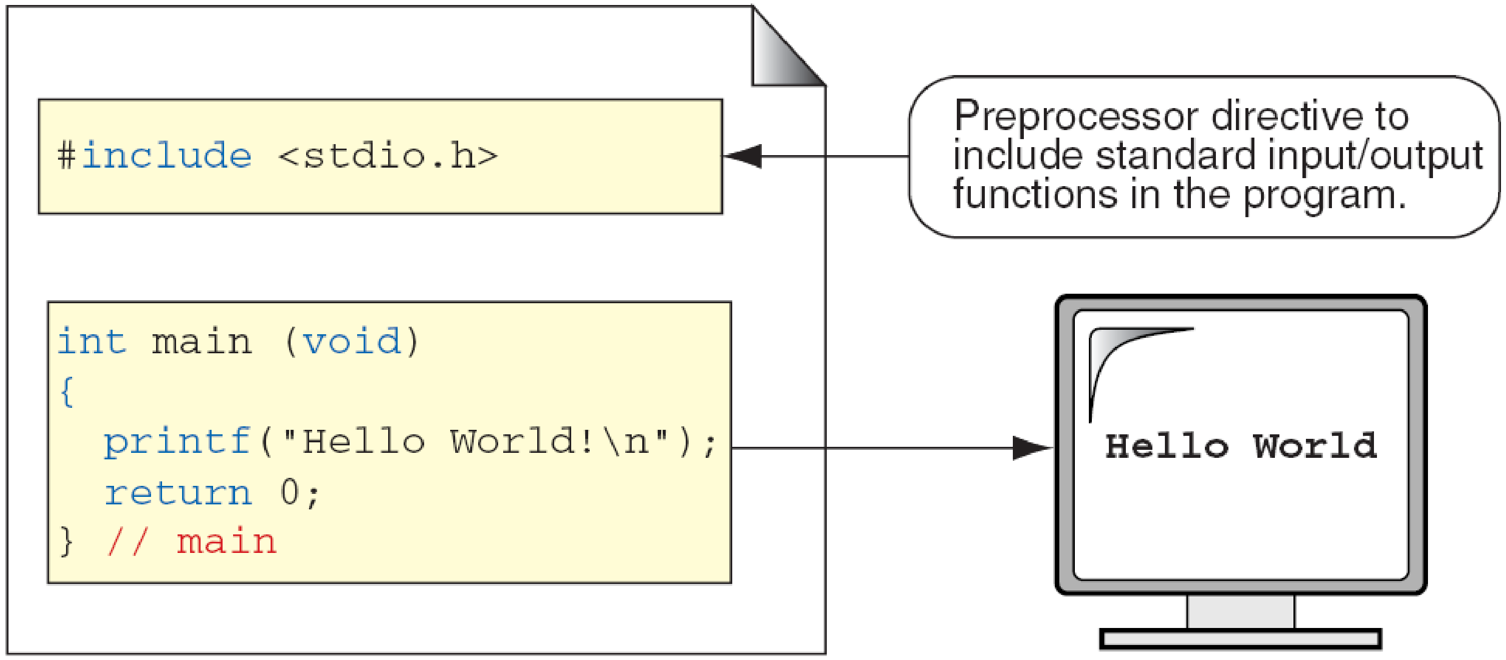
…………..

Function n

(User defined functions)

1-3 Basic Structure of 'C' Program

- **Structure of C Program**



```
#include <stdio.h>

int main (void)
{
  printf("Hello World!\n");
  return 0;
} // main
```

Preprocessor directive to include standard input/output functions in the program.
Hello World

return 0 – successful execution

1-3 Basic Structure of ‘C’ Program Continued..

Pre-Processor Directive :

- Pre-Processor Directives tells the compiler to include Input/Output information for your program.
- Preprocessor directives start with #

#include copies a file into the source code

#include <systemFilename>

Eg.

```
 #include<stdio.h>    // Standard Input/Output
#include<math.h>   //Advanced Math Function(eg.Trignometry)
#include<conio.h>   // Console Input / Output
#include<stdlib.h>  //  Arithmetic and Sorting & Searching
```

The #define directive

- It is a preprocessor compiler directive and not a statement., and therefore it should not end with a semicolon
- Symbolic constants are generally written in uppercase so that they are distinguished from lowercase variable names.
- Placed generally before the main() function.

Eg. #define PI 3.14

1-3 Basic Structure of ‘C’ Program Continued..

Global Declaration

- A global variable is a variable that is declared outside all functions.
- A global variable can be used in all functions.

Eg.
```
#include<stdio.h>
int a;   } //Global Declaration
int b;
int main()
{
        int Sum;
        Sum=add();
        printf(“Answer : %d”,Sum);
        return 0;
}
int add ()
{
        return a+b;     // As Declared Global can used in all function
}
```

1-3 Basic Structure of ‘C’ Program Continued..

The main() function :

- Programs must have a main() function.
- Allowed Format :

1. main()
2. main(void)
3. int main()
4. int main(void)
5. void main()
6. void main(void)

1-3 Basic Structure of ‘C’ Program Continued..

Local Definition :

- A **local** variable can only be used in the function where it is declared.

Eg.

```
int main()
{
        int a,b,c; // Local Declaration
        printf(“Hello”);
        return 0;
}
int add()
{
        return a+b;   // Cannot accessible outside main() function as declared
                              as  Local
}
```

1-3 Basic Structure of ‘C’ Program Continued..

printf() function :

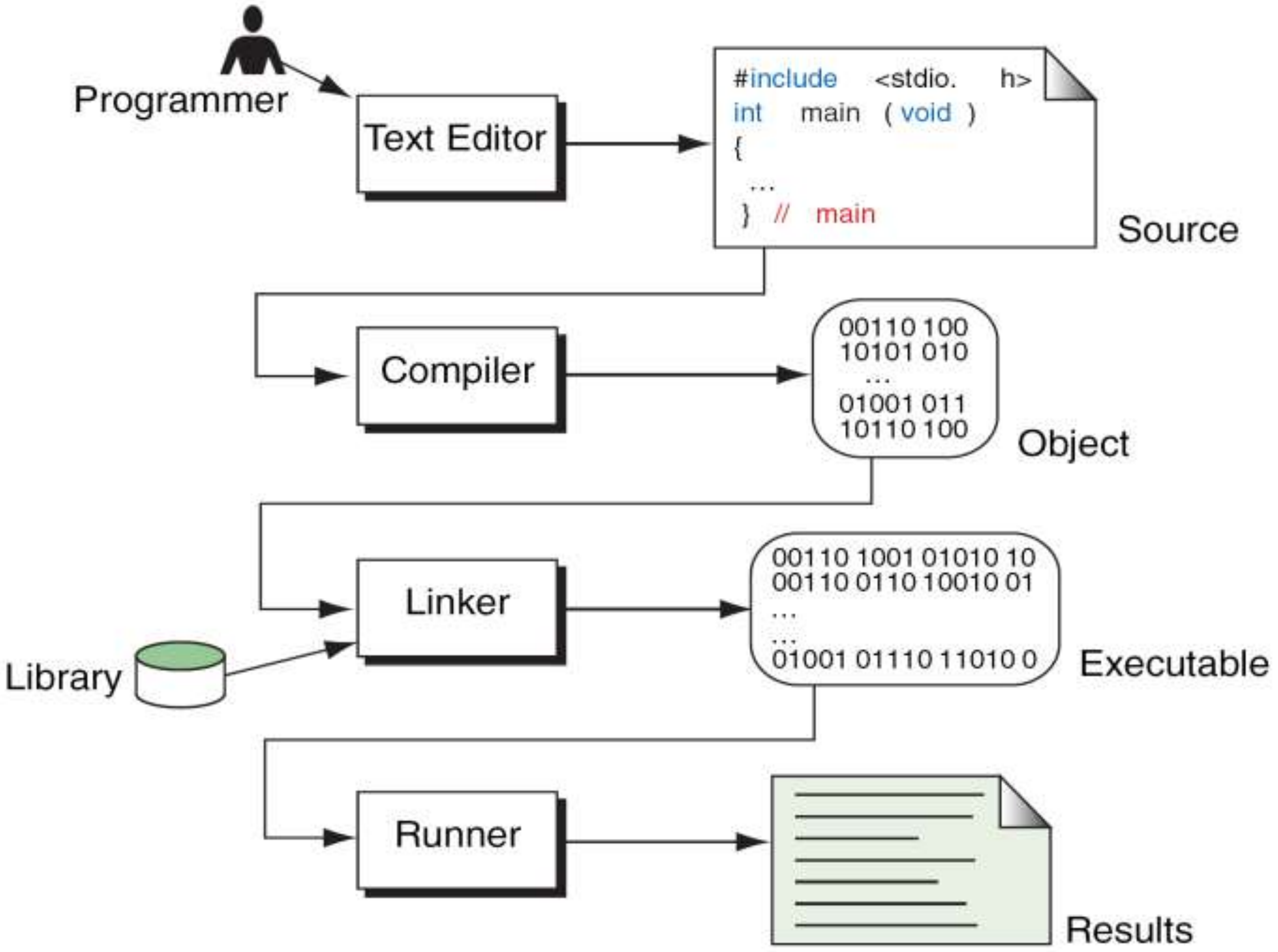
- It is used to print standard output (Screen).
- It can take variable number of argument.
- The first argument must be a String.

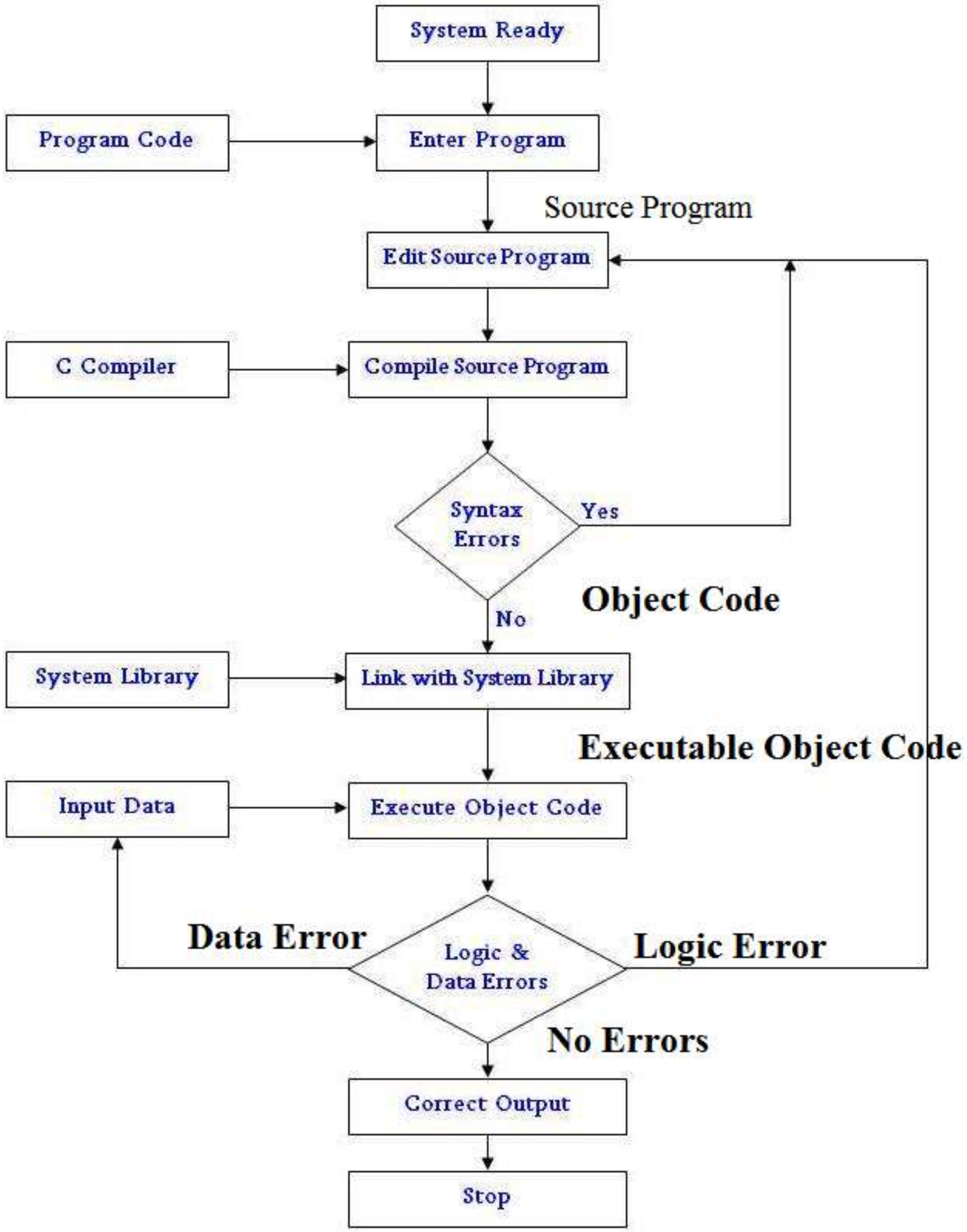
Eg.

printf(“Message”, Variable or Variable List);

- printf(“Hello”);
- Printf(“Answer : %d”,Sum);

1-4 Compilation of C Program




System Ready
Program Code
Enter Program
Source Program
Edit Source Program
C Compiler
Compile Source Program
Syntax Errors
Yes
No
Object Code
System Library
Link with System Library
Executable Object Code
Input Data
Execute Object Code
Data Error
Logic & Data Errors
Logic Error
No Errors
Correct Output
Stop

[illegible]

1-4 Compilation of C Program Continued…

Pre-Processor

- Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing.
- Commands used in preprocessor are called preprocessor directives and they begin with "#" symbol.
- The main function of the C preprocessor is to remove comments from the source code and interpret preprocessor directives which are given by the statements that begin with #.
- The #include statement can either be called with

  #include<file.h>

  or with

  #include "file.h"
- The first method tells the preprocessor to look for the file in the standard include directories.
- The second method, which uses the quotes, tells the preprocessor that the file to be included is in the local directory.

1-4 Compilation of C Program Continued…

Compiler

- The compiler translates the C code into assembly language, which is a machine level code that contains instructions.
- Converted assembly language code having extension ".s".

Assembler

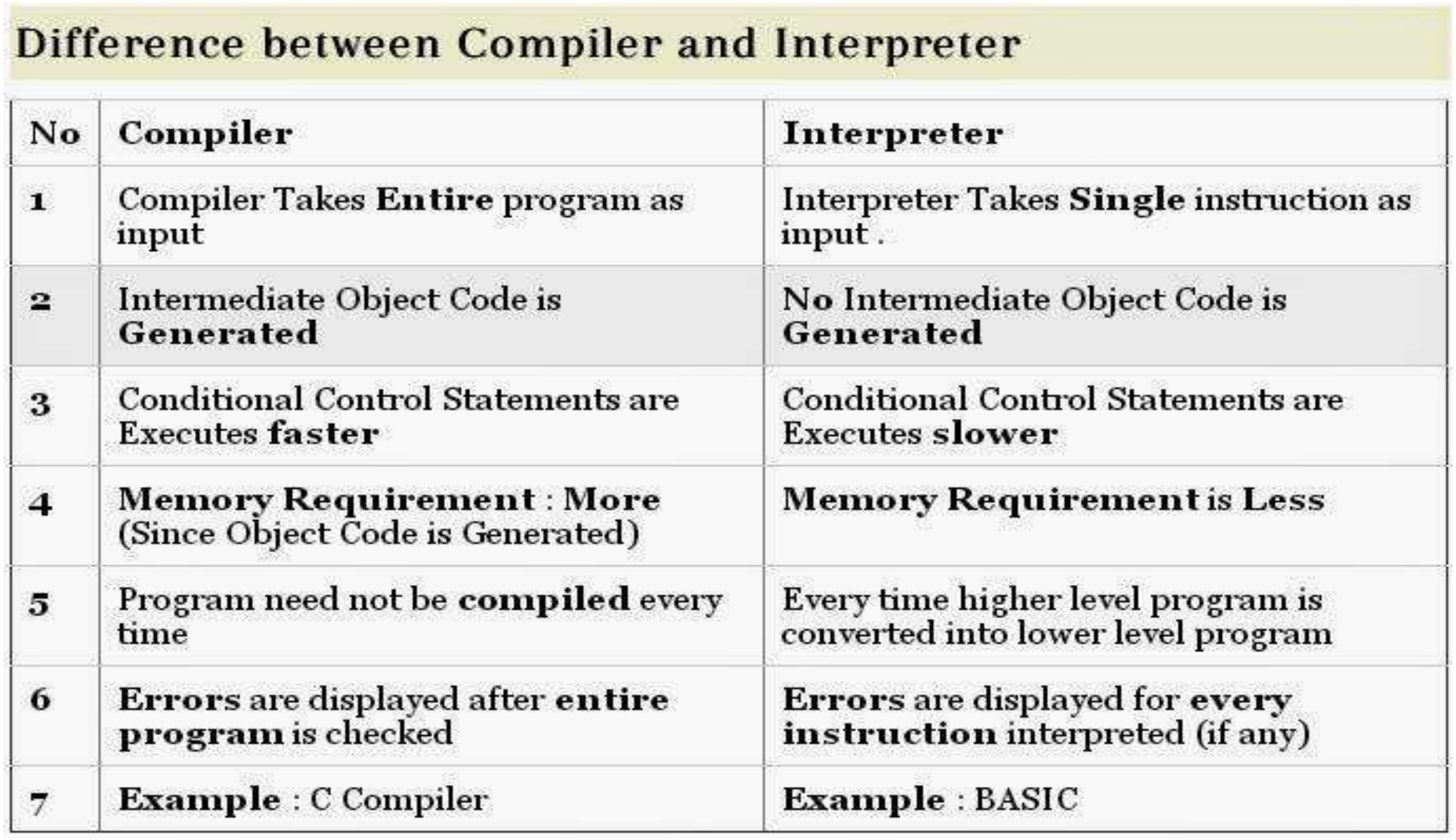
- Assembler translates the assembly language code into machine language.
- Converted code having extension ".o" (Object file)

1-4 Compilation of C Program Continued…

Linker

- The object file **hello.o** contains a binary version of the machine language that was created from your source code hello.c. In order to create the executable hello or a.out, you need to use the linker to process your main function.
- In example, we used the printf function. The printf function is a standard function that is provided by the C compiler that your current object file knows nothing about.
- In order to use this function, we need to use the linker in order to link our program with the precompiled libraries provided to us by the C compiler. The linker links other precompiled object files or libraries together and creates the executable hello.

1-3 Compiler & Interpreter

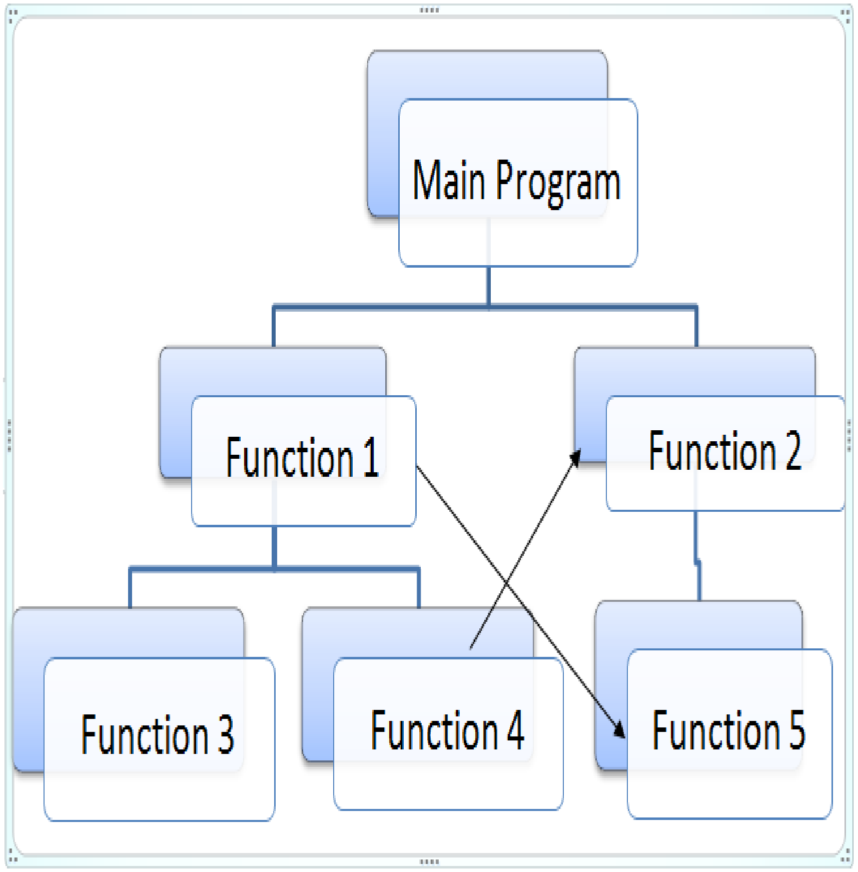


Difference between Compiler and Interpreter

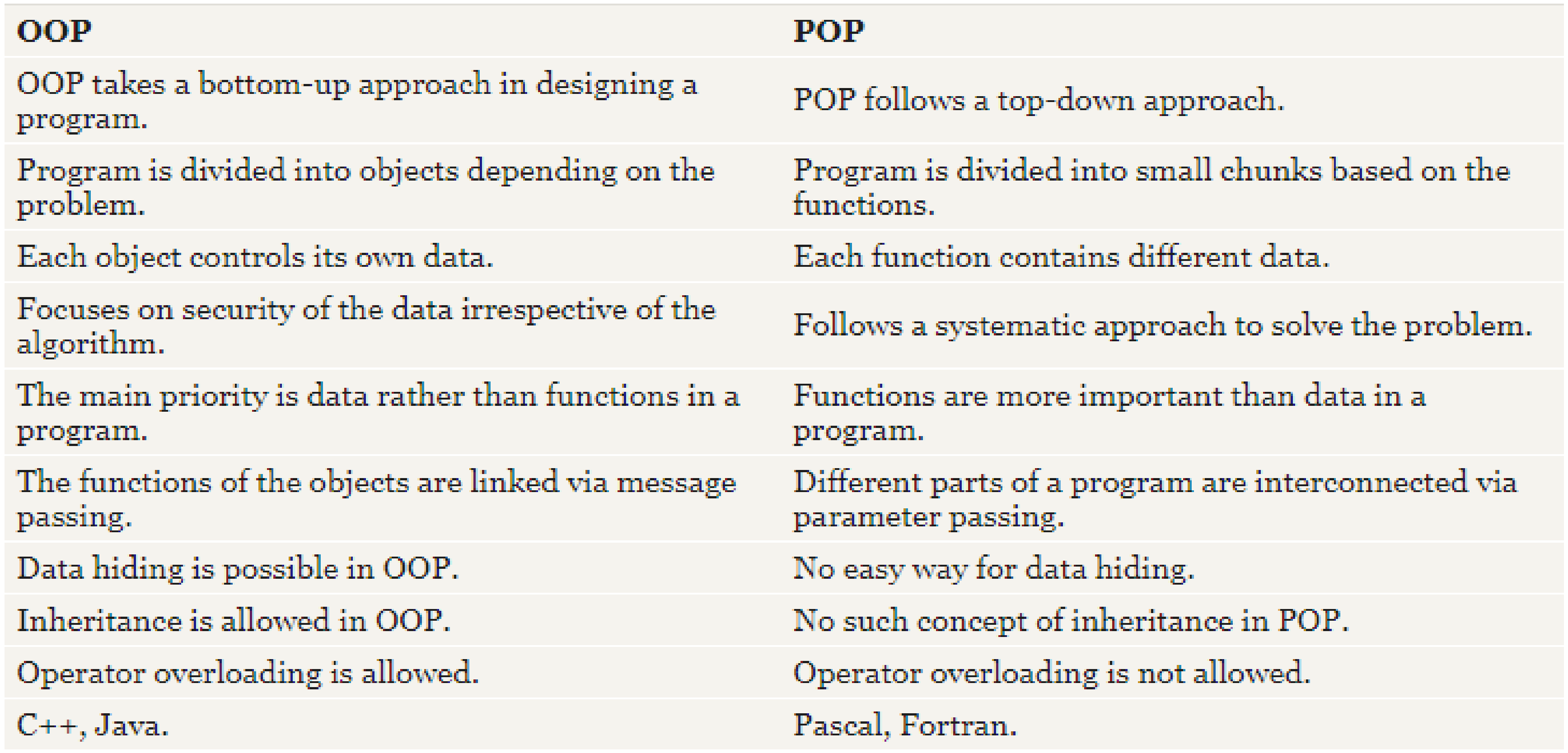
No	Compiler	Interpreter
1	Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input .
2	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement** : **More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
5	Program need not be **compiled** every time	Every time higher level program is converted into lower level program
6	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
7	**Example** : C Compiler	**Example** : BASIC

1-3 Procedural Oriented Programming (POP)

- A program in a procedural language is a list of instruction where each statement tells the computer to do something. It focuses on procedure (function) & algorithm is needed to perform the derived computation.
- When program become larger, it is divided into function & each function has clearly defined purpose. Dividing the program into functions & module is one of the cornerstones of structured programming.
- E.g.:- c, basic, FORTRAN.



1-3 Difference between Object Oriented Programming(OOP) and Procedural Oriented Programming (POP) Continued…



OOP	POP
OOP takes a bottom-up approach in designing a program.	POP follows a top-down approach.
Program is divided into objects depending on the problem.	Program is divided into small chunks based on the functions.
Each object controls its own data.	Each function contains different data.
Focuses on security of the data irrespective of the algorithm.	Follows a systematic approach to solve the problem.
The main priority is data rather than functions in a program.	Functions are more important than data in a program.
The functions of the objects are linked via message passing.	Different parts of a program are interconnected via parameter passing.
Data hiding is possible in OOP.	No easy way for data hiding.
Inheritance is allowed in OOP.	No such concept of inheritance in POP.
Operator overloading is allowed.	Operator overloading is not allowed.
C++, Java.	Pascal, Fortran.

Questions from previous year papers

1. What is the significance of Compiler?
(2 Marks)
2. Which are the Pre-processer directive? (2 Marks)
3. Define any two chracteristics of C Language with example. (3 Marks)
4. What is the purpose of Linker?(2 Marks)

Questions from previous year papers

Fill in the blanks

1. C was developed by _________
2. _______ is a group of C statements that are executed together.
3. Execution of the C program begins at ________.
4. In memory, characters are stored as _________.
5. The statement *return 0*; returns 0 to the __________.
6. A C program ends with a ____________.

Questions from previous year papers

State True or False

1. We can have only one function in a C program.
2. Header files are used to store the program's source code.
3. Stdio.h is used to store the source code of the program.
4. The closing brace of the main() is the logical end of the program.
5. The declaration section gives instruction to the computer.
6. Declaration of the variables can be done anywhere in the program.

Questions from previous year papers

Review Questions

1. Explain the difference between declaration and definition.
2. Give the structure of C program.
3. Why do we include <stdio.h> in our programs?
4. Explain the utility of #define and #include statements.
5. Draw and explain the process of compiling and running a C program.
6. Write the various forms of main that are allowed.
7. Draw a flowchart describing the history of C.
8. Explain the importance of C (in points).